**paper 1**

**setup**

* The bottle capper machine, for instance, normally made a rhythmic sound. When something went wrong with the capper, the rhythm changed thus alerting the user.
* added sound to a programming environment [I]. By attaching a different sound to each routine or block of code, semantic errors, like infinite loops, were easily detected
* When given a choice between a text based application and a windows based one, visually impaired users prefer the text based product since it is easier for them to use.

**How it works**

task trial with each of the audio environments, the subject would continually "hear" the cursor's location By pressing and releasing the mouse button consecutively an audible comparison is made so that the subject can determine the similarity between the two sounds, and hence the relationship

between the cursor and the target.

**Results**

**In this figure larger subjective results are better. The tonal sound environment was rated as the best sound environment over the musical and orchestral sound environments on all four of the subjective measures by visually impaired users.**

**paper 2**

**Related work**

**activity theory**

Despite promising results, the application-centric model remains largely a siloed experience, where applications and the computational

entities that they generate have little awareness of each other, leaving the user to develop individual strategies for coordination

**Organizing by activity**

Humans do substantial work to translate their activities into tasks that can be completed within the hierarchical structure of a computing environment. For example, when copying materials from one place to another. The students encountered numerous challenges as they painstakingly attempted to complete the task. First, locating the appropriate parent directory, creating and naming a new folder, and relocating a file to the desired location must all be completed using keyboard commands to interface with multiple parts of the file system: operating system, file browser, and application. The non visual have to orient themselves first Rather than remember the keyboard commands to save or move a file, they must first cognitively orient to the context.

**Activity tracking**

An object that played an important role at the start of an activity may never be used again. Similarly, some objects may not find utility until the final steps leading to activity completion. Managing these variations throughout the life of an activity is supported in the traditional desktop through visually oriented design cues such as recognition and spatial arrangement, and virtual desktops.

**Operationalizing Actions**

At first, mouse use might be action oriented, where a novice user consciously interacts with its various buttons and controls. Eventually, through practice, mouse use becomes operationalized, moving from a conscious to unconscious operational state. By restructuring the application-document metaphor, activity-based systems are effectively adding an additional layer of complexity that users must navigate.

**KlnD**

**Activity management**

Activity tokens (see Figure 5) are tangible icons that represent an activity. right). Transferring the representation of activity to the physical world has the advantage of introducing the effects of unconscious operation

***Interaction History***

KInD is situated between the intentions of the user and the execution of those intents in the computer system. Events are logged in the host computer’s file system to provide a history of interaction, and associations between activity token and entities. As system entities are opened they become an active part of the activity. When they are closed, they move out of the activity space, but remain in its history.

**Contextual change**

To accommodate the movement across activities as well as their associated entities, KInD functions within different contexts.

**Modeling Interactions on Need**

we describe how an activity-centered nonvisual system should consider user needs, tasks, and interactions to reduce complexity in a nonvisual computing environment. The Activity Tokens and underlying API in KInD support user need by organizing interrelated computational entities into high-level activities. Activity Tokens can be interchanged as user needs shift, keeping the working environment isolated from unrelated entities.

**Supporting System Change.**

The KInD system supports change in two ways. First, by providing tactilely unique representations of activities through Activity Tokens and activity context through Virtual Docks and the Context Dial, users can directly recall system state.

**Supporting User Goals**

accompanies tasks goals for nonvisual users by replacing multiple keyboard commands (many of which varys across applications [8]) with tangible interactions.